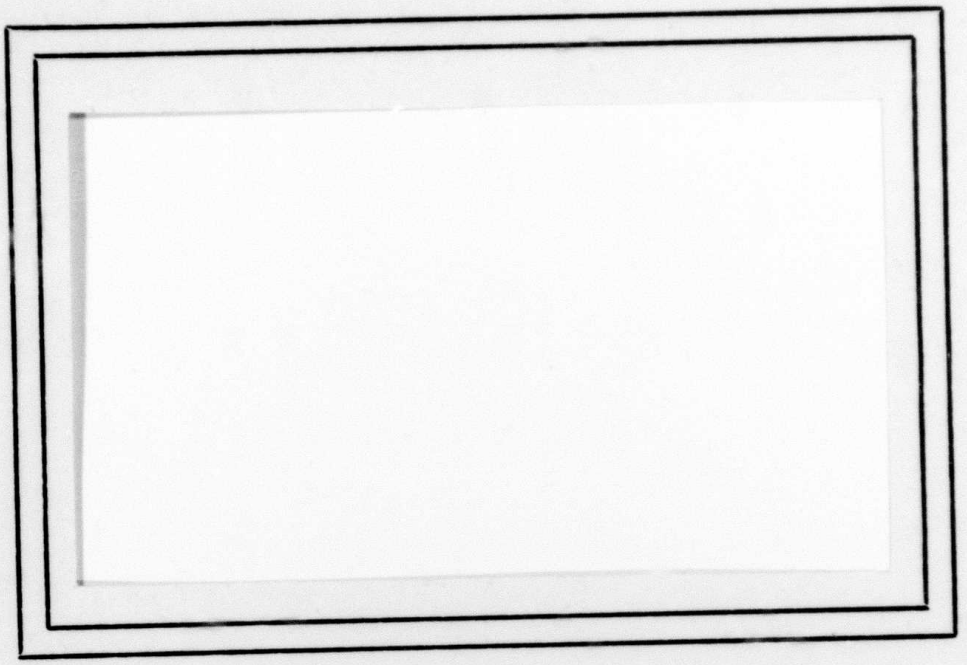
LEVEL II

1

AD A109557



COMPUTER SCIENCE TECHNICAL REPORT SERIES

DTIC ELECTE JAN 12 1982 S D E

UNIVERSITY OF MARYLAND
COLLEGE PARK, MARYLAND
20742

DTIC FILE COPY

This document has been approved for public release and sale; its distribution is unlimited.

82 01 12 062

LEVEL II

1

TR-888
DAAG-53-76C-0138

April, 1980

CLUSTERING OF COLLINEAR LINE SEGMENTS

Ann Scher
Michael Shneier
Azriel Rosenfeld

Computer Vision Laboratory
Computer Science Center
University of Maryland
College Park, MD 20742

Technical rept.

ABSTRACT

A number of methods are presented for finding clusters in collinear collections of line segments. The methods are of two kinds--merging methods and splitting methods. Both make use of an evaluation function, and several alternative functions are illustrated. The methods are evaluated using randomly generated clusters on backgrounds containing varying amounts of noise.

DTIC
ELECTE
JAN 12 1982
S D
E

The support of the U.S. Army Night Vision Laboratory and the Defense Advanced Research Projects Agency under Contract DAAG-53-76C-0138 (DARPA Order 3206) is gratefully acknowledged.

This document has been approved for public release and sale; its distribution is unlimited.

411074

1. Introduction

The problem of detecting clusters in collinear collections of line segments is of interest both in psychology and in computer vision. For example, the result of running an edge detector over an image is usually a set of line segments, some of which correspond to the same edge but are broken because of noise in the image. It would be useful to be able to cluster these lines, but, at the same time, to avoid including spurious or distant responses in the cluster.

Proximity grouping is one of the problems that has long concerned Gestalt psychologists [2,3,4]. Surprisingly little is known about the mechanisms involved, however, although Wertheimer [3], working with dot clusters, observed that the lengths of the gaps between dots were insufficient to define groupings. A ratio between intra-cluster and inter-cluster gap lengths should be used instead. This observation, suitably modified for the case of line segments, forms the basis for some of the methods to be discussed here.

This paper treats a one-dimensional case of proximity grouping. Indeed, it deals with the most general binary one-dimensional case. Several methods for clustering collinear line segments are presented. The algorithms fall into two main classes, merging methods and splitting methods. A further method adapted from an algorithm for finding peaks in waveforms is also presented.

The merging methods initially assume that every individual line segment forms a trivial "cluster". Each cluster attempts to expand by including neighboring segments, until an evaluation function prevents further additions. The methods may be sequential, for example starting at one end of the data and moving towards the other, or they may operate in parallel on all the segments in the collection. By iterating a clustering process it is possible to find clusters of clusters to any desired level.

In the splitting methods, the whole collection of line segments is initially treated as forming a single cluster. This cluster may be split recursively into subclusters based on the separations between segments in the cluster. Various evaluation functions are investigated for controlling the splitting process.

All the evaluation functions rely on ratios between the length of a line segment and the sizes of the gaps surrounding it. Various ways of combining these measures are discussed below.

Accession For	
NTIS GRA&I	X
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A	

1.1 The Data

Clustering is a complex process that depends on several different kinds of information. Two important kinds are proximity and similarity. In this study we were concerned with proximity grouping, and in the experiments we attempted to remove the effects of similarity grouping. This was done by allowing the gap sizes to vary, but keeping the size of the line segments fixed.

The data were generated using two random processes with different means and variances. One of the processes generated a single cluster with mean gap size m_c, and variance v_c. The other process was used to generate the background, and had mean $m_b > m_c$, and variance $v_b > v_c$. The background was generated in two parts, one to the left of the cluster and one to the right. This was done to eliminate the chance of two line segments partly intersecting within the cluster, which would have given rise to line segments of non-standard length.

Six sets of data were generated for the examples to be presented here (Figure 1). In all of them the same central cluster was used, with mean gap size 10 and variance 3. The background in the examples was initially generated with a mean gap size of 40 and a variance of 13. These parameters were gradually made to approach those of the central cluster by changing the mean and variance as shown in Table 1.

2. Descriptions of the Algorithms

2.1 Merging Methods

All the merging methods, and some of the splitting methods, make use of the following evaluation function. This function is based on the observation that the ratio between the length of a group of segments and the amount of black (i.e. the proportion of the group that is not gap) is a useful measure for grouping.

Denote the sequence of line segments and gaps that forms the data by $g_0, a_0, g_1, a_1, \ldots, g_n, a_n, g_{n+1}$, where the g_i denote lengths of gaps, and the a_i denote lengths of line segments. g_0 and g_{n+1} are usually taken to be zero.

Then define

$$f(g_i) = f(a_{i-1}, a_i) =$$

$$\frac{\sum_{j=0}^{i} a_j}{\sum_{j=0}^{i-1} a_j} \cdot \left\{ \frac{\sum_{j=0}^{i} a_j}{\sum_{j=0}^{i} (g_j + a_j)} \Big/ \frac{\sum_{j=0}^{i-1} a_j}{\sum_{j=0}^{i-1} (g_j + a_j)} \right\} \qquad (1)$$

This equation has the form

$$f = \frac{\text{new amount of black}}{\text{old amount of black}} \cdot \frac{\text{new ratio of black to length}}{\text{old ratio of black to length}}$$

Notice that the first part of the expression (i.e. the ratio of the amounts of black) is always greater than 1, since the amount of black always increases. The second part, however, varies between 0 and 1 depending on the relative sizes of the gaps and the line segments. The first part of the expression reflects a desire to increase the number of segments in the

cluster, while the second penalizes the addition of new line segments if this involves adding a large gap as well. The expression thus has sensible properties for evaluating clusters. It has shown itself robust and useful in a variety of clustering methods.

Method 1

The first and simplest merging method involves a left to right scan of the data, applying the evaluation function at each point as follows.

Given the sequence $g_0, a_0, g_1, a_1, \ldots, g_n, a_n, g_{n+1}$, where g_0 and g_{n+1} are zero,

1. Set CLUSTER = $\{a_0\}$

   Set i = 1

2. If i = n+1 stop

   If $f(g_i) \geq 1$ then

   Set CLUSTER = CLUSTER $\cup$ $\{a_{i-1}\}$

   Set i = i+1

   Repeat Step 2

   else output CLUSTER as a complete cluster

   Set CLUSTER = $\{a_i\}$

   Set i = i+1

   Repeat Step 2, disregarding $g_0, a_0, \ldots, g_{i-1}$ in the calculation of f (i.e. assuming a_i is the start of the data)

This method finds many "noise" clusters as well as the "real" clusters (Figure 2). The following method is an attempt to decrease the number of noise clusters that are reported.

Method 2

Method 2 is similar to Method 1, except that, in addition to running the evaluation from left to right, it is run from right to left also. A substantial improvement over Method 1 is obtained by intersecting the results of these two passes (Figure 3).

Method 3

Methods 1 and 2 are essentially sequential in nature. A more satisfying algorithm is one that can operate on all segments in parallel. Method 3 uses local neighborhoods of each segment, and joins segments to their neighbors if the evaluation criterion is met.

For every sequence a_i, g_{i+1}, a_{i+1}, compute (in parallel)

$$f(a_i, a_{i+1}) = \frac{a_i + a_{i+1}}{a_i} \cdot \frac{a_i + a_{i+1}}{a_i + g_{i+1} + a_{i+1}}$$

and $f(a_{i+1}, a_i)$, defined similarly (these are not the same because of the division in the first term).

If $f(a_i, a_{i+1}) \geq 1$ and $f(a_{i+1}, a_i) \geq 1$, join a_i and a_{i+1} into a cluster; otherwise leave them separate.

Notice that the function f is the same as (1), except that it is restricted to individual pairs of segments. Despite the local nature of the evaluation, this method works reasonably well (Figure 4).

2.2 Splitting Methods

With the exception of the first method described, all the splitting methods have the following basic structure.

1. Calculate a best splitting point according to some criterion. If there is no such point, stop.
2. Split the line segments into two subsets at the calculated point, and discard the subset that is less dense. Density is defined as the amount of black in the subset, divided by its total length. The splitting point is assumed to lie in the center of a gap, so that half the gap length is counted in each density calculation.
3. Repeat steps 1 and 2 on the denser part.
4. Evaluate the clusters that were found and filter out those considered to be noise.

Method 4

The simplest splitting scheme for cluster detection proceeds as follows.

1. Calculate the evaluation function (1) for each gap.
2. Find the minimum value of the function, and split if the value is less than 1.
3. Recursively apply steps 1 and 2 to the left and right sides of the split, stopping if the minimum value of (1) is greater than or equal to one, or when only single segments remain.

Simple splitting methods like this one have two problems (see Figure 5). First, they find spurious "noise" clusters in addition to the "correct" clusters. Second, they do not find the borders of clusters accurately. Both these problems are caused by an inflexible splitting function. In the first case, the splitting function becomes more and more short-sighted as the recursion proceeds until it acts like the local merging function above. The second problem is due to the fixed threshold in the splitting function. The methods to be discussed below attempt to overcome these problems by taking a more global view of the data and by using more sophisticated evaluation techniques.

The succeeding methods follow the basic format described above. The main differences between the methods are in the functions used to find the splitting point at each stage, and in their stopping criteria. All the methods rely on an evaluation step which determines the optimal cluster from among all the clusters found.

Method 5

Given a sequence $g_i, a_i, g_{i+1}, a_{i+1}, g_{i+2}$, define

$$F = \frac{a_i + a_{i+1}}{a_i} \cdot \frac{\frac{a_i + a_{i+1}}{g_i + a_i + 2*g_{i+1} + a_{i+1} + g_{i+2}}}{\frac{a_i}{g_i + a_i + g_{i+1}}} \qquad (2)$$

This function has the same format as (1). The first term is the ratio of the amount of black in the pair of lines to that in the single segment. The other term is similar to the old ratio of black to length, but includes the gap lengths on either side of the segments, thus making the function non-symmetric. The function (2) is defined only for pairs of line segments.

At each gap in the data, (2) is applied twice, once as $F(a_i,a_{i+1})$, and once as $F(a_{i+1},a_i)$. The score is taken as a weighted average of $F(a_i,a_{i+1})$ and $F(a_{i+1},a_i)$, where $F(a_i,a_{i+1})$ is weighted by $g_i+a_i+g_{i+1}$, and $F(a_{i+1},a_i)$ is weighted by $g_{i+1}+a_{i+1}+g_{i+2}$. That is, each term is weighted by the sum of the lengths of the segment and the gaps surrounding it. These weighted averages have their maximum change on the inside edge of a cluster. That is, the actual splitting point is one gap out from where the weighted average changes most. This over-shooting appears to be caused by the edge effects inherent in splitting methods.

A ranking is assigned to each cluster found. It is the change between the score described above and the score of the cluster found one level down in the recursion (a sub-cluster of the current cluster). The ranks are used to order the clusters. For instance, if only one cluster is desired, the one with the smallest rank is chosen, i.e. that cluster which is most similar to its sub-cluster.

Figure 6 shows the result of applying Method 5 to the test data. Figure 7 shows some results of substituting Function (1) for Function (2) in the method. Function (2) was found to be more effective in this method than Function (1). This is partly because it has a slightly more global view of the data (an extra gap length), and partly because of the asymmetry. It is to be expected that Function (2) would improve the performance of Methods 1-4 as well. Function (1) is, however, simpler to compute, and need only be applied once at each gap.

Method 6

Method 6 is slightly more global than the previous method. It uses a neighborhood to calculate an evaluation function that works as an edge detector, or gradient operator. The size of the neighborhood is that of the largest gap in the data. This means that the neighborhood over which the gradient is calculated could end in the middle of a segment. It is thus not possible to treat the data in terms of segments and gaps as in the other methods. This method can be considered an approximation to Method 10 described below.

At each endpoint, the absolute difference between the sum of the values in the neighborhood to the left of the point and that to the right of the point is used as the score for splitting at that point. The point with the highest score is chosen.

Note that it is only necessary to consider endpoints of segments, since the maximum gradient will always occur there. The point at which the split is actually made is the central point of the gap. As usual, this method is repeated on the denser side of the gap.

As in the previous method, a rank is assigned to each cluster found. It is the change between the score for the cluster and the score of its sub-cluster. The cluster with the smallest rank is chosen. Figure 8 shows the results of applying the method to the data of Figure 1.

Method 7

The function used in Method 7 is based on examining average gap lengths. At each potential splitting point (i.e. each gap between segments) the average of all gap lengths to the left of the point is calculated, as is that for the gap lengths to the right. (This method is global since it considers all of the data at every point.) The split point chosen is the one that has the greatest negative difference between its left average and the preceding point's left average or the greatest positive difference between its right average and the preceding point's right average. The assumption is that the average gap size on the left of the cluster has its sharpest decrease when the point is just inside the cluster, and similarly, its sharpest

increase just after the right of the cluster. The splitting stops when there is no decrease in average gap size on the left or increase in average gap size on the right, or the point chosen is one of the endpoints of the data.

Note that half the gap at which the split is being evaluated is added in to the average for each side.

The method usually provides a good estimate of the correct split point, but, due to edge effects and the small size of some neighborhoods, it sometimes over- or undershoots the best split point. A further examination of the local neighborhood about the split point is thus made to locally improve the positioning.

Figure 9 shows the situation for the right side of a cluster (i.e. the left region is denser than the region to the right of the split point). The situation is analogous for the left side of a cluster. For this case, the four gaps A, B, C, and D are examined, and max(D-C,C-B,B-A) is found. If D-C is the maximum value, then D is a bigger gap than C, and the split point is moved to D. If B-A is the maximum, the split point is moved to B; otherwise it remains at C. With this fine tuning, the true edges of the cluster can be found more accurately.

As before, a rank is applied to each cluster found. It is the difference between the cluster's average gap size and the average gap size of its sub-cluster. The cluster with the smallest rank is chosen. Figure 10 shows the results of applying the method to the data of Figure 1.

Method 8

Method 8 is similar to Method 7 but uses the variance of the gap sizes on either side of each gap instead of the average of the gap sizes to calculate the splitting point. The assumption is made that the cluster has a mean and variance different from the background, and that an endpoint of the cluster can be found by looking for changes in the variance to the left and right of each gap. The split point that is chosen is that which has the greatest positive difference between its left variance and the previous point's left variance or the greatest negative difference between its right variance and the previous point's right variance. This method works well if the background is fairly uniform. It would break down if the background were too noisy, or if its variance were too close to that of the cluster.

A cluster is ranked by the difference between its variance and the variance of its sub-cluster. The cluster with the smallest rank is chosen. Figure 11 shows the results obtained from applying this method to the test data.

Method 9

The function used for splitting in Method 9 is based on entropy. For a group of segments $A = a_1, a_2, \ldots, a_n$, the entropy of A is defined as

$$\text{entropy}(A) = \sum_i p_i \log(p_i)$$

The p_i's are defined by the gaps on either side of each segment a_i as:

$$p_i = 1 - \frac{2 * \text{smallest gap in data}}{\text{length of left gap} + \text{length of right gap}}$$

The p_i's are then normalized.

Segments in the interior of a cluster tend to have smaller gaps separating them than those in the background. The p_i's are large if the gaps surrounding the segment are large, and small if the gaps are small. They thus approximate the probability that segment a_1 is not in a cluster. Similarly, entropy(A) estimates the likelihood that the group of segments A is entirely in the background.

The gap with the greatest entropy on either its left or right side is chosen as the split point, since it is most likely to delimit the foreground from the background of a cluster.

A cluster is ranked by the difference of this greatest entropy and the greatest entropy of its sub-cluster. The cluster with the lowest rank is chosen. Figure 12 shows the clusters found using this method.

2.3 Finding Peaks in a Waveform

In addition to the splitting and merging methods, another method was developed. This method was originally designed to find peaks and valleys in waveforms (Eklundh and Rosenfeld [1]), but was adapted to detect clusters of line segments. The main differences between the waveform application and the clustering application are that the waveform data change less abruptly than clustering data, and can take on more values. Also, in finding clusters, only the peaks (or only the valleys) are of interest, whereas in waveform applications both must be found.

Method 10

The method of Eklundh and Rosenfeld was adapted to search only for peaks. Like Method 6, this method treats the data as a sequence of points or pixels. It involves applying simple difference operators at each point to neighborhoods having a wide range of sizes. Simple comparisons between the outputs of these operators for various sizes and positions are used to evaluate the results. This method is expensive because of the number of neighborhoods to be compared, and it works no better than some of the methods discussed earlier (Figure 13).

3. Discussion

One result that is apparent from the experiments described above is the usefulness of the evaluation function (1). This function was shown to be robust and applicable to several different merging and splitting algorithms. Note that the function is not restricted to the case in which all the segment lengths are the same, but that when it is applied to this case, it has a somewhat simpler form.

Some general remarks can be made about the methods described in Section 2. First, it is apparent that the simple merging algorithms are reasonably powerful, but have the characteristic of finding many small "noise" clusters in addition to the "real" clusters. This is partly because the so-called noise clusters do indeed group together, but can be largely ascribed to the local nature of the merging processes.

Splitting algorithms have a more global view of the data, but require more sophisticated evaluation techniques to find the clusters. It was found that evaluation methods as simple as those used for merging did not give satisfactory results (e.g., Method 4). This means that the splitting methods described here are more expensive than the merging methods, but they do give cleaner output.

A major disadvantage of using splitting algorithms is the introduction of edge effects into the evaluation. Whereas merging can be done locally, and is too myopic to be affected

by the endpoints of the data, the splitting algorithms always have to deal with the edges explicitly at each stage in the splitting process. Various methods of avoiding edge effects were used. For example, the data were treated as having gaps on either side (Method 9), or the points on the border were ignored (Method 5), or the data was reflected about the border (Method 10). All of these expedients are reasonable, but may affect the results produced by the evaluation function. This is apparent in some of the methods that required fine tuning of the split position.

In practice, the splitting methods are preferable to the merging methods. All the splitting methods work well with the exception of Method 4, which suffers from similar defects to the merging methods. Of the remaining methods, Method 9 based on entropy was less successful than the others because of the poor evaluation function. The best overall method to use is probably Method 5. It is computationally simpler than the other methods, and works just as well.

It would be of interest to consider using a splitting technique to find initial candidates for clusters, and then applying a less expensive merging technique to delimit the clusters more exactly. Although from the examples shown here such a technique appears redundant because of the success of the splitting methods in finding clusters, such an approach might have its merits for more complex data.

4. Conclusions

A number of methods have been described for finding proximity groupings of line segments. It has been shown that a particular evaluation function (1) is useful for finding clusters using a number of different methods. This function is based on the proportion of a grouping that is made up of line segments, and the total length of the grouping, including the gaps.

It has further been shown that simple merging techniques can successfully find clusters, although the local nature of such methods gives rise to somewhat noisy output. While simple splitting algorithms are not as successful, more sophisticated techniques have the advantage of a global view of the data, and can be employed to find the clusters without being affected as much by the noise.

References

1. J. O. Eklundh and A. Rosenfeld, Peak detection using difference operators, IEEE Trans. PAMI, vol. 1, pp. 317-325, 1979.

2. K. Koffka, Principles of Gestalt Psychology, Harcourt, Brace, New York, 1935.

3. M. Wertheimer, Unterschungen zur lehre von der Gestalt II, Psychologische Forschung, Zeitschrift fur Psychologie und ihre Grenzwissenschaften, vol. 4, pp. 301-350, 1923.

4. L. Zusne, Visual Perception of Form, Academic Press, New York, 1970.

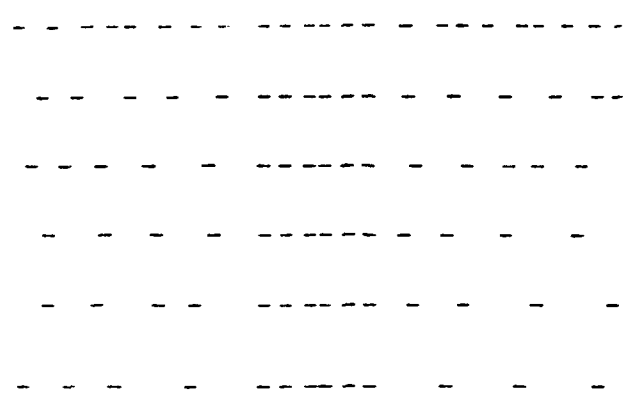


Figure 1

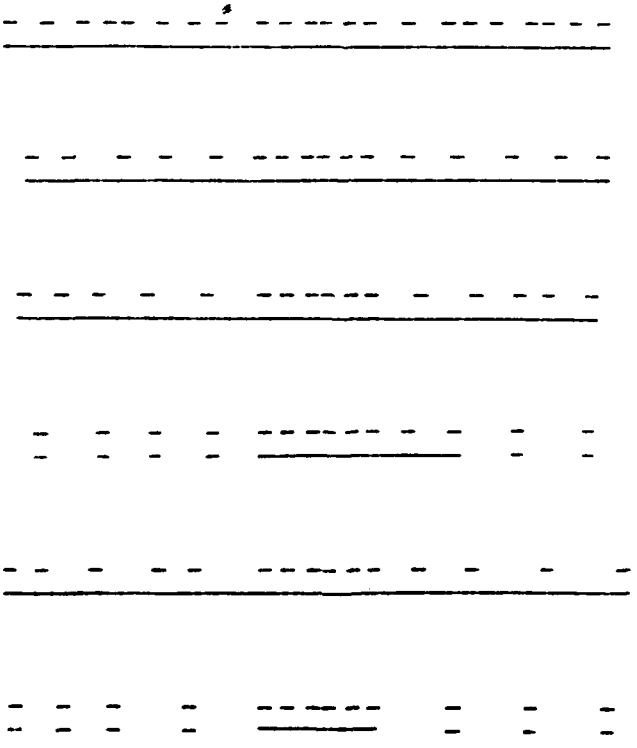


Figure 2

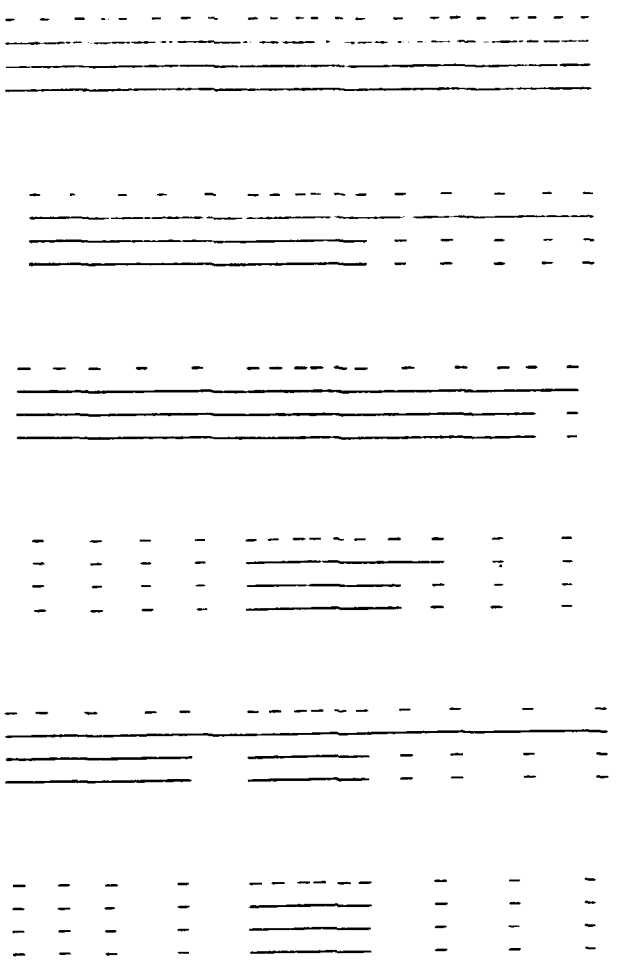


Figure 3

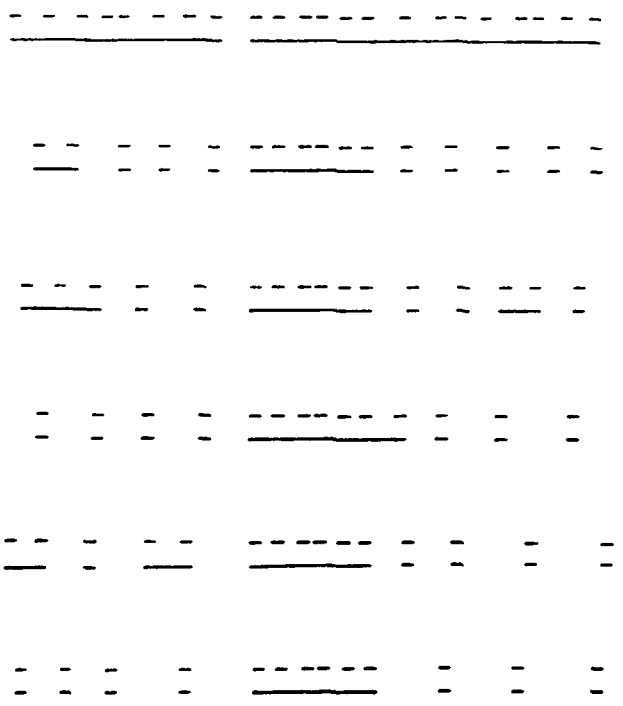


Figure 4

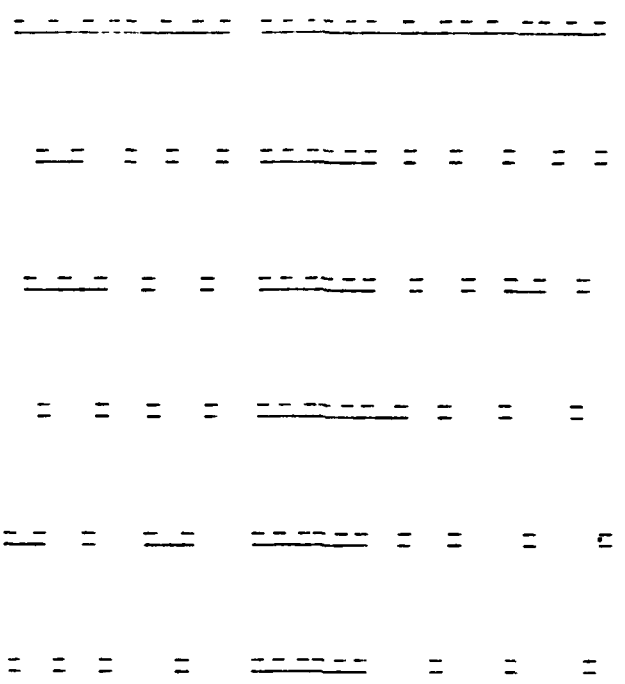


Figure 5

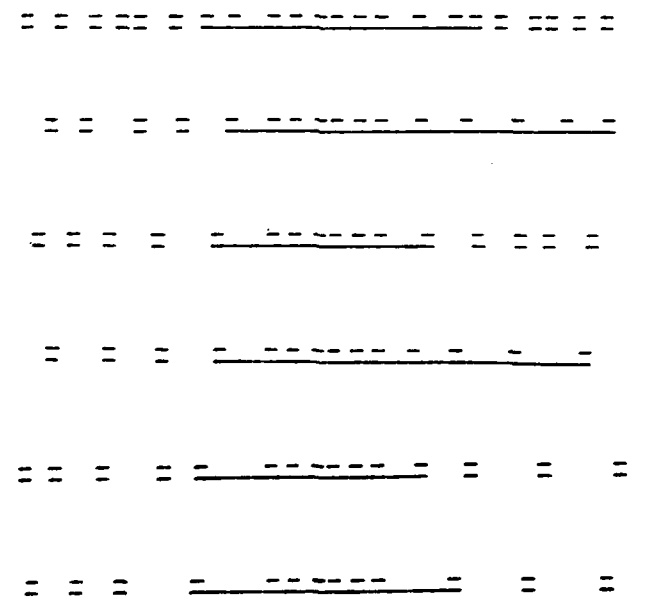


Figure 6

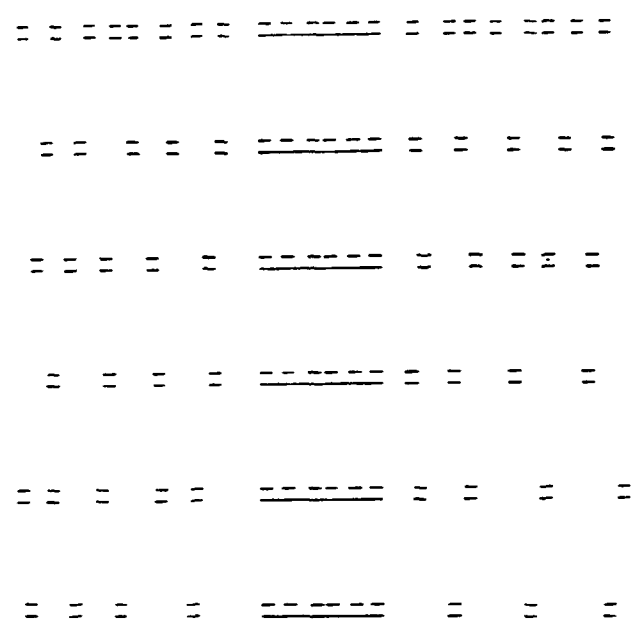


Figure 7

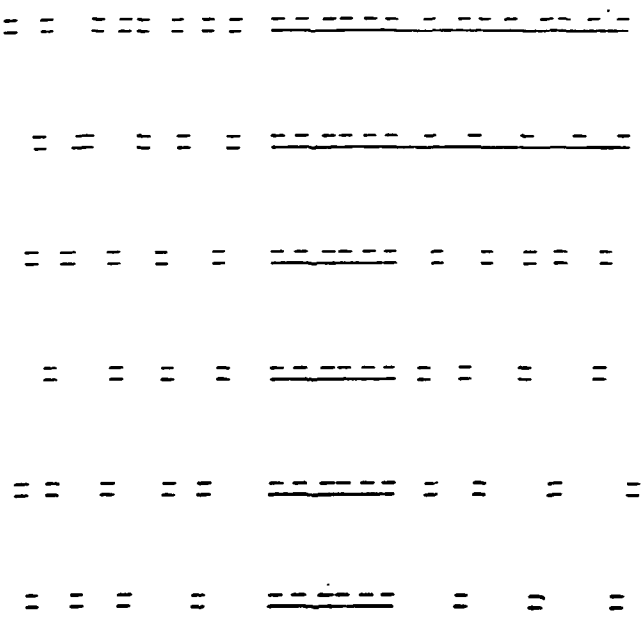


Figure 8

Figure 9

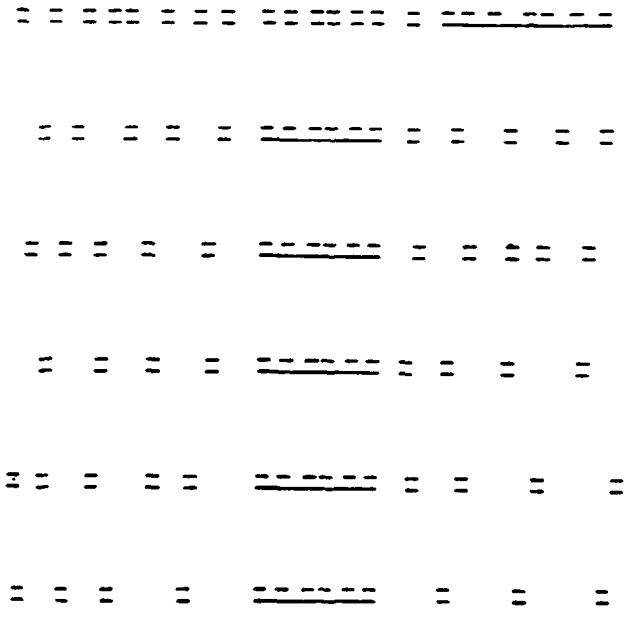


Figure 10

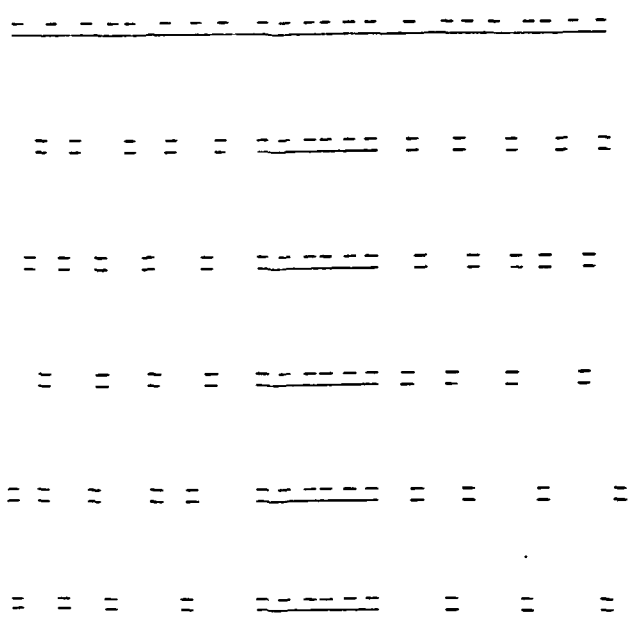


Figure 11

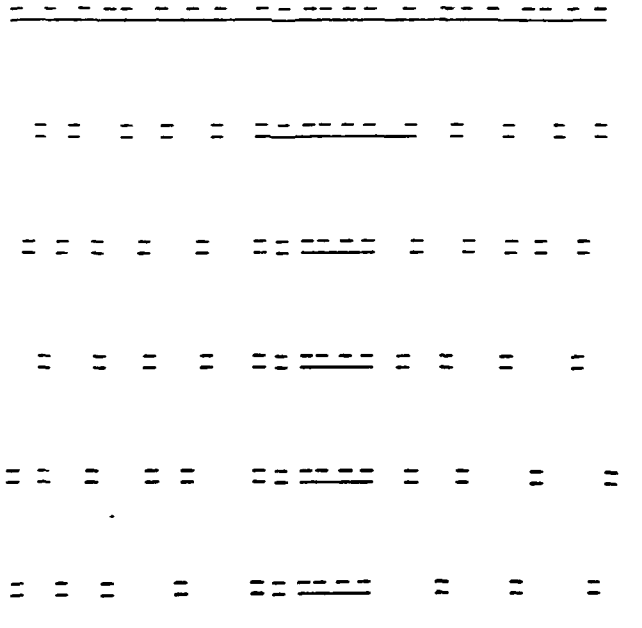


Figure 12

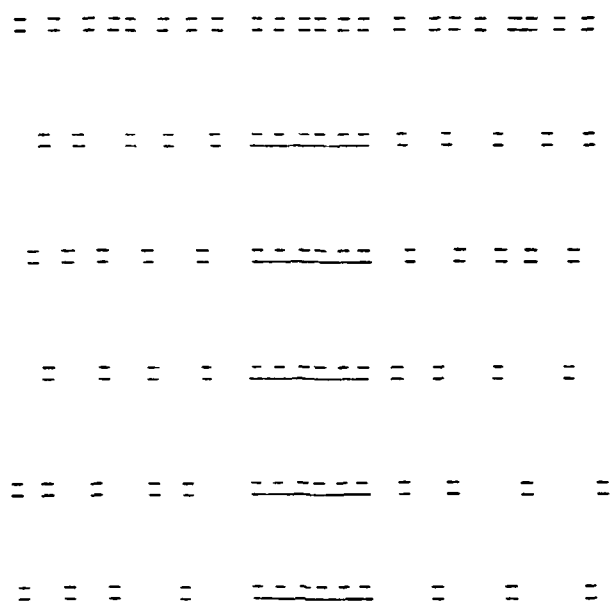


Figure 13

Data set	Mean	Variance
Cluster	10	3
1	15	5
2	20	7
3	25	8
4	30	10
5	35	12
6	40	13

Table 1

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A084 551	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) CLUSTERING OF COLLINEAR LINE SEGMENTS		5. TYPE OF REPORT & PERIOD COVERED Technical
		6. PERFORMING ORG. REPORT NUMBER TR-888
7. AUTHOR(s) Ann Scher, Michael Shneier, Azriel Rosenfeld		8. CONTRACT OR GRANT NUMBER(s) DAAG-53-76C-0138
9. PERFORMING ORGANIZATION NAME AND ADDRESS Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD 20742		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS U.S. Army Night Vision Laboratory Ft. Belvoir, VA 22060		12. REPORT DATE April, 1980
		13. NUMBER OF PAGES 28
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Image processing
Pattern recognition
Clustering
Collinearity
Visual perception
Proximity

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A number of methods are presented for finding clusters in collinear collections of line segments. The methods are of two kinds--merging methods and splitting methods. Both make use of an evaluation function, and several alternative functions are illustrated. The methods are evaluated using randomly generated clusters on backgrounds containing varying amounts of noise.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)